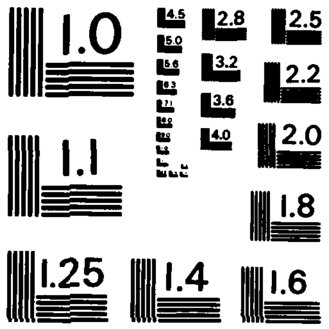
AD-A145 786 STATISTICAL ANALYSIS OF SOME TRAVELING SALESMAN ALGORITHMS(U) CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENCES RESEARCH GROUP E BALAS ET AL. 1/1

UNCLASSIFIED JUN 84 MSRR-501 N00014-82-K-0329 F/G 12/1 NL

END

FILMED

DTIC

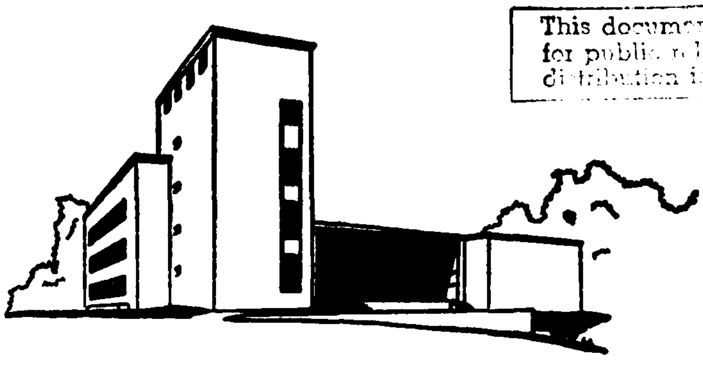

1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



1

Management Science Research Report No. MSRR 501

STATISTICAL ANALYSIS OF SOME TRAVELING SALESMAN ALGORITHMS

by

Egon Balas
Carnegie-Mellon University

Timothy W. McGuire
Carnegie-Mellon University

and

Paolo Toth
University of Bologna

June, 1984

This work was supported for the first author by Grant ECS-8205425 of the National Science Foundation and Contract N00014-82-K-0329 NR 047-607 with the U.S. Office of Naval Research, and for the third author by the Consiglio Nazionale delle Ricerche of Italy. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

Management Science Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

This paper reports the results of a statistical analysis of the performance of three branch and bound algorithms for the general (asymmetric) traveling salesman problem on randomly generated test problems with up to 325 cities. Three types of functions, polynomial, superpolynomial (log-exponential) and exponential, were fitted to the performance data of each of the algorithms by least squares techniques. The three functions describe almost equally well the behavior of the algorithms in the range of problem sizes examined.

Key words: Traveling salesman problem, branch and bound, algorithm performance, statistical analysis

A-1

STATISTICAL ANALYSIS OF SOME TRAVELING SALESMAN ALGORITHMS

by

Egon Balas
Carnegie-Mellon University

Timothy W. McGuire
Carnegie-Mellon University

and

Paolo Toth
University of Bologna

1. Introduction

Given n cities and a nonnegative cost c_{ij} of traveling from city i to city j, the traveling salesman problem (TSP) asks for a minimum cost tour of the n cities. In graph-theoretical terms, the n cities are the nodes of a complete directed graph G = (N,A) and an optimal tour is a minimum cost directed Hamilton cycle. The TSP is called symmetric if $c_{ij} = c_{ji}$ for all i,j, asymmetric otherwise. By associating a variable x_{ij} with every arc, defined to be 1 if (i,j) is an arc of the tour (Hamilton cycle) to be constructed and 0 otherwise, one can write the TSP (see [5]) as

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}x_{ij} \tag{1.1}$$

subject to

$$\left\{\begin{array}{ll} \sum_{j=1}^{n} x_{ij} = 1 , & i = 1,\dots,n \\ \sum_{i=1}^{n} x_{ij} = 1 , & j = 1,\dots,n \end{array}\right. \tag{1.2}$$

$$x_{ij} = 0 \text{ or } 1 , \qquad i,j = 1,\dots,n \tag{1.3}$$

$$\sum_{i\in S} \sum_{j\in S} x_{ij} \le |S|-1, \quad \forall\, S\subset\{1,\dots,n\},\ 2 \le |S| \le n-1. \tag{1.4}$$

Without (1.4), the above problem is the $n \times n$ assignment problem (AP), equivalent to the linear program (1.1), (1.2) and

(1.3′) $\quad x_{ij} \geq 0, \quad i,j = 1,\ldots,n.$

A solution to AP is either a tour or a collection of subtours (directed cycles of length less than n), and the role of the inequalities (1.4) (called subtour elimination constraints) is precisely to exclude the occurrence of subtours in a solution. On the other hand, the presence of (1.4) also requires that of (1.3), since the polyhedron defined by (1.2), (1.3′) and (1.4) has fractional vertices. Thus the TSP is an integer program.

Although it has a very special structure, the TSP still has the reputation of a notoriously difficult combinatorial problem. It is known to be NP-complete, and hence in all likelihood there exists no polynomial time algorithm for solving it - in the worst case sense, i.e., in the sense of being guaranteed to solve every instance of the TSP. This, however, leaves open the question concerning the expected or average time required to solve a randomly chosen TSP. Whether the expected running time of various TSP algorithms is an exponential or a polynomial function of n, or perhaps something in between like a superpolynomial or log-exponential function, is at present an open question. A probabilistic analysis of the problem, although very tempting, seems difficult. We say it is tempting, since the AP, one of the most common relaxations of the TSP, is solvable in $O(n^3)$ time in general, and in $O(n^2)$ time when processed repeatedly in the context of a branch and bound procedure for the TSP. At the same time, the AP has n! solutions (assignments), of which (n-1)! are TSP solutions (tours). Furthermore, in the context of the TSP only those assignments are of interest that contain no diagonal elements (of the assignment tableau, i.e., satisfy $x_{ii} = 0$, $i = 1,\ldots,n$), and while the assignment problem with this extra constraint is as easy to solve as the original one, the number of its solutions is $n!/e$ rounded to the nearest integer, where e is the base of the natural logarithm (see, for instance, [6], p. 10). Thus on the average one in every n/e "diagonal-free" assignments is a tour. This, together with the fact that a breadth-first branch and bound procedure can rank the k best diagonal-free assignments by solving at most kn assignment problems, hence by a computational effort of $O(kn^3)$, suggests that a procedure of this type might be able to find an optimal tour in a TSP with randomly generated costs by an average computational effort of $O(n^4)$. For this to be true, however, the probability distribution of the costs would have to be such that, roughly speaking, if all assignments are listed in order of increasing costs, tours are evenly distributed throughout the list (rather than "clustered" in certain parts of it). At this point it is not clear whether this is the case for any distribution of the costs. Several attempts at a probabilistic analysis essentially based on the above considerations have either argued that this is very likely to be the case (see [3], and also the letter [7] pointing out the inadequacy of the argument), or have simply assumed it to be true [9]. Needless to say, this does not settle the problem.

While the theoretical issue remains unresolved, it seems of considerable interest to examine from this point of view the empirical performance of some of the more efficient TSP algorithms on problems with randomly generated costs. With this in mind, we have undertaken a statistical analysis of the performance of three well known AP-based TSP algorithms, as reported in the literature by their authors. To be more specific, we fitted various types of approximating functions to the published performance data for each algorithm, in an attempt to decide which type of function describes best the relationship between problem size as defined by n, and solution time. Our aim in reporting these results is not to compare the algorithms, but to compare the performance of the different approximating functions that we tested.

2. The Algorithms

The three algorithms that we examined are those of Smith, Srinivasan and Thompson [10], Carpaneto and Toth [4], and Balas and Christofides [1], to be denoted in the following by SST, CT and BC, respectively. All three are enumerative procedures using the AP (with forbidden diagonal elements) as a relaxation of the TSP. However, the SST and CT algorithms use the AP with the original objective function (1.1), whereas the BC procedure uses a Lagrangean constructed by taking into the objective function, with appropriate multipliers, selected inequalities of the type (1.4), or inequalities derived from (1.2) and (1.4). Whatever the objective function of a problem P, we denote its optimal value by v(P).

All three procedures are initialized by putting the TSP on the list of active subproblems, and each of them stops when the list of active subproblems is exhausted. The subproblems on the list differ from the original TSP (and from each other) by specified subsets of arcs that are forcibly included (I) into, or excluded (E) from the solution.

In each of the three algorithms, at a typical iteration some or all of the following steps may be executed:

Subproblem Selection. Choose a subproblem TSP_i from the list according to some rule.

The SST algorithm uses the rule known as depth first or LIFO (last in first out). This consists of always choosing one of the subproblems (nodes of the search tree) generated at the last branching step, in the order of nondecreasing lower bound (as given by the AP solution value plus penalties associated with each node); and when no more such subproblems exist, backtracking to the parent node and applying the same rule to its brothers. This rule has the advantage of modest storage requirements and easy bookkeeping. Its disadvantage is that possible erroneous decisions (concerning arc inclusion or exclusion) made early in the procedure cannot be reversed until late in the procedure.

The subproblem selection rule used by the CT algorithm is known as breadth first. It consists of always choosing the subproblem with the best (smallest) lower bound. This rule has the desirable feature of keeping the search tree as small as possible, but on the other hand it requires considerable storage space.

The BC algorithm uses a combination of the depth first and breadth first rules: a successor of the current subproblem is selected whenever available; otherwise a node with best (smallest) evaluation E is chosen from the list, where E is based on the value of the lower bound, corrected for the "distance" of the subproblem solution from a tour.

Having chosen a subproblem from the list, the SST and BC algorithms go to Lower Bounding, whereas the CT procedure goes to Branching.

<u>Branching</u>. Break up the feasible set of TSP_i by augmenting the sets I_i and E_i of included and excluded arcs, respectively.

The SST algorithm chooses a minimum cardinality subtour in the current solution to AP_i, whose free arc set is, say, $\{(i_1,j_1),\ldots,(i_t,j_t)\}$, and creates t successors of node i, say i1,...,it, by defining

$$\begin{array}{lll} & E_{i1} = E_i \cup \{i_1,j_1)\}, & I_{i1} = I_i \\ (2.1) & \text{- - - - - - - - - - - - - - - - -} & \\ & E_{it} = E_i \cup \{(i_t,j_t)\}, & I_{it} = I_i \cup \{(i_1,j_1),\ldots,(i_{t-1},j_{t-1})\}. \end{array}$$

The CT algorithm uses the same rule (2.1), but chooses for branching a subtour whose set of free arcs is of minimum cardinality.

The BC algorithm uses intermittently two branching rules, one of which is (2.1), while the other one is derived from the principle of conditional bounds, which can be stated as follows. Let H be the arc set of the current best tour, let $\hat{c}_{ij} \geq 0$, $i,j = 1,\ldots,n$, be a set of reduced costs associated with the linear program (1.1), (1.2), (1.3′), (1.4), and let $S \subset H$, $S = \{(i_1,j_1),\ldots,(i_p,j_p)\}$ be such that

$$\sum_{(i,j)\in S} \hat{c}_{ij} \geq U-L(\hat{c}),$$

where U is the current upper bound, and $L(\hat{c})$ is the lower bound associated with the cost vector $\hat{c}$.

Further, let Q_r, $r = 1,\ldots,p$ be arc sets of G satisfying

$$\sum_{r|(i,j)\in Q_r} \hat{c}_{i_r j_r} \leq \hat{c}_{ij} \quad , \quad \forall\ (i,j)\in A.$$

Then every tour of value less than U satisfies the disjunction

$$x_{ij} = 0,\ (i,j)\in Q_1 \vee\ldots\vee x_{ij} = 0,\ (i,j)\in Q_p.$$

This disjunction can be used to create p successors of node i, by defining

$$\begin{array}{lll} & E_{i1} = E_i \cup Q_1 \ , & I_{i1} = I_i \\ (2.2) & \text{- - - - - - - - - - - - -} & \\ & E_{ip} = E_i \cup Q_p \ , & I_{ip} = I_i\ . \end{array}$$

The rules (2.1) and (2.2) are used intermittently, since their relative strengths may differ at different nodes of the search tree. The choice is based on a coefficient of relative strength.

After branching, the SST and BC algorithms place the newly generated subproblems on the list, and go to Subproblem Selection. The CT algorithm instead goes to Lower Bounding.

Lower Bounding. Calculate a lower bound on $v(TSP_i)$. This is known to be the crucial ingredient of any branch and bound method, in that the stronger the bounds that are derived, the fewer subproblems have to be examined.

The SST algorithm generates the lower bound $L_i = v(AP_i)$ by solving AP_i when TSP_i is selected from the list. If $L_i \geq U$ or the solution to AP_i is a tour, the algorithm returns to Subproblem Selection, otherwise it goes to Branching. After branching it also calculates an estimated lower bound for the new nodes generated, by adding to $v(AP_i)$ the penalty associated with the arc exclusions and inclusions prescribed by the branching rule.

The CT algorithm solves the assignment problem for each node generated in the Branching step as soon as it was generated, and places on the list only those successors of TSP_i for which the lower bound obtained is below the current upper bound U. If any of the solutions to the new assignment problems is a tour of improved value, U is updated. The algorithm then returns to Subproblem Selection.

The BC algorithm uses an elaborate lower bounding procedure that goes well beyond solving the assignment problem. If the optimal solution $\bar{x}$ to the latter defines a tour, U is updated and the algorithm returns to Subproblem Selection. Otherwise, a Lagrangean function is constructed by combining the original objective function with a family of valid inequalities (mostly facets of the TSP polyhedron) derived from (1.4) and (1.2), with multipliers chosen in such a way that the optimal solution $\bar{x}$ to the assignment problem remains optimal for the new objective function. If the inequalities to be used are written in the generic form

$$\sum_{i \in N} \sum_{j \in N} a^t_{ij} x_{ij} \geq a^t_o \quad , \quad t \in T$$

for some index set T, then the Lagrangean is

$$L(w) = \min_{x \in X} \sum_{i \in N} \sum_{j \in N} (c_{ij} - \sum_{t \in T} w_t a^t_{ij}) x_{ij} + \sum_{t \in T} w_t a^t_o ,$$

where X is the assignment polytope and w the vector of multipliers. L(w) provides a valid lower bound for any vector of $w \geq 0$. The strongest bound is obviously given by the Lagrangean dual $\max\{L(w): w \geq 0\}$, which however is difficult to solve in the given instance. The BC algorithm replaces the Lagrangean dual with the restricted Lagrangean problem

$$\text{(RL)} \qquad \max \left\{ L(w) \;\middle|\; \begin{array}{l} w \geq 0 \\ \\ u_i + v_j + \sum_{t \in T} w_t a^t_{ij} \begin{cases} = c_{ij} & \text{if } \bar{x}_{ij} = 1 \\ \leq c_{ij} & \text{if } \bar{x}_{ij} = 0 \end{cases} \end{array} \right\}$$

and finds a good approximate solution to (RL). In particular, it uses a sequence of polynomial time algorithms (bounding procedures) for identifying new inequalities that can be added to L(w) with a positive multiplier w_t without changing the other components of w.

When the sequence of lower bounding procedures is exhausted, the BC goes to Upper Bounding.

Upper Bounding. The SST and CT algorithms do not use special upper bounding procedures, but generate upper bounds as a byproduct of solving AP: whenever the solution is a tour, it provides such a bound.

In the BC algorithm, the Lagrangean function L(w), besides providing a strong lower bound, also plays the role of defining an "admissible subgraph" $G_o = (N,A_o)$, whose arc set is

$$A_o = \{(i,j)\epsilon A | u_i+v_j+\sum_{t\epsilon T} w_t a_{ij}^t = c_{ij}\}.$$

The inclusion of each new inequality into L(w) adds at least one new arc to the set A_o, and when the lower bounding procedures are exhausted, G_o is strongly connected and without articulation points. The BC algorithm then uses a specialized implicit enumeration procedure with a cut-off rule in an attempt to find a tour in G_o. If a tour H is found, U is updated. Furthermore, by construction G_o has the property that if all inequalities with $w_t > 0$ are tight for H, then H is optimal for the current subproblem TSP_i. In this case the algorithm returns to Subproblem Selection; otherwise it removes from G all arcs (i,j) whose reduced costs exceed U-L(w) (i.e., fixes at 0 the corresponding variables) and goes to Branching.

The main characteristics of the three algorithms as implemented by their authors are summarized in Table 1.

3. The Data

Performance data for solving the asymmetric TSP are reported in [10], [4] and [1], for FORTRAN implementations of the SST, CT and BC algorithms, respectively (see also [2]), and are reproduced for convenience in Table 2. Each of the three codes was run on (different) sets of asymmetric TSP's whose costs were drawn independently from the discrete uniform distribution on $\{i: i = 1,2,...,1000\}$. The data consist of the arithmetic average computing time (in seconds) and arithmetic average number of nodes of the search tree. This latter number refers to all nodes generated in the case of the CT and

Table 1. Synopsis of the three TSP algorithms

	SST	CT	BC
Relaxation	AP with TSP objective	AP with TSP objective	AP with Lagrangian objective
Lower bounding	$v(AP_i)$, obtained by parametric simplex method, plus penalties	$v(AP_i)$, obtained by Hungarian method (post-optimizing version)	lower bound on $v(RL)$, obtained by polynomial-time approximation procedures
Branching rules	(2.1)	(2.1)	(2.1) and (2.2)
Subproblem selection	depth first	breadth first	depth first upon forward step, breadth first upon backtracking
Upper bounding	no special procedure	no special procedure	tour-finding heuristic applied to admissible graph
Variable fixing ("reduction")	no	no	yes

Table 2. Computational results on randomly generated asymmetric TSP's

n	Nodes of the search tree			Computing time (seconds)		
	SST(1)	CT(2)	BC(2)	SST(3)	CT(4)	BC(5)
40	26	27	-	2.9	0.9	
50	11	-	12	1.7	-	0.2
60	39	24	-	9.3	2.2	-
70	32	-	-	8.5	-	-
75	-	-	27	-	-	0.3
80	32	42	-	13.8	6.6	-
90	82	-	-	42.0	-	-
100	87	56	39	53.0	10.4	0.7
110	24	-	-	22.3	-	-
120	65	61	-	62.9	16.2	-
125	-	-	43	-	-	1.1
130	97	-	-	110.1	-	-
140	130	57	-	165.2	18.7	-
150	50	-	46	65.3	-	2.0
160	70	73	-	108.5	32.8	-
170	98	-	-	169.8	-	-
175	-	-	58	-	-	4.2
180	215	69	-	441.4	28.8	-
200	-	58	63	-	35.7	6.1
220	-	43	-	-	46.7	-
225	-	-	84	-	-	10.4
240	-	63	-	-	53.4	-
250	-	-	89	-	-	13.7
275	-	-	106	-	-	21.7
300	-	-	124	-	-	38.4
325	-	-	142	-	-	49.7

(1) Number of nodes that were explored; (2) total number of nodes; (3) UNIVAC 1108; (4) CDC 6600; (5) CDC 7600

BC algorithms, but only to the nodes selected and explored in the case of SST. The averages are based on 5, 20, and 10 replications for each problem size for SST, CT, and BC, respectively. The SST algorithm solved problems of sizes 40, 50,...,180; CT solved problems of sizes 40, 60,...,240; and BC solved sizes 50, 75,...,325. The computations were performed on a UNIVAC 1108, CDC 6600, and CDC 7600 for the SST, CT, and BC algorithms, respectively. We note that the CDC 7600 is approximately three times faster than the CDC 6600 or the UNIVAC 1108 (which are of roughly equal speed).

The natural logarithms of the average solution times for the three algorithms are plotted against the natural logarithm of problem size in Figure 1. A straight line relationship would suggest that solution time is a polynomial function of problem size. The SST curve is the least smooth of the three, undoubtedly because each point is an average of only five trials. The BC points exhibit the least variance around a smooth function, although they are averages of only ten trials whereas the CT points are averages of twenty trials.

We also have solution time data for each of the twenty trials for each of the eleven problem sizes for the CT algorithm. Table 3 presents some relevant summary statistics for these data. The medians are peculiarly insensitive to problem size for $160 \leq n \leq 240$. Hence the increases in the arithmetic and geometric means for those problem sizes are due to the right tail of the distribution, which is corroborated by the 75th percentile solution times. Overall, the various summary statistics for the distribution of solution times are highly correlated (see Table 4). The standard deviations of the solution times increase strongly with problem size; even the standard

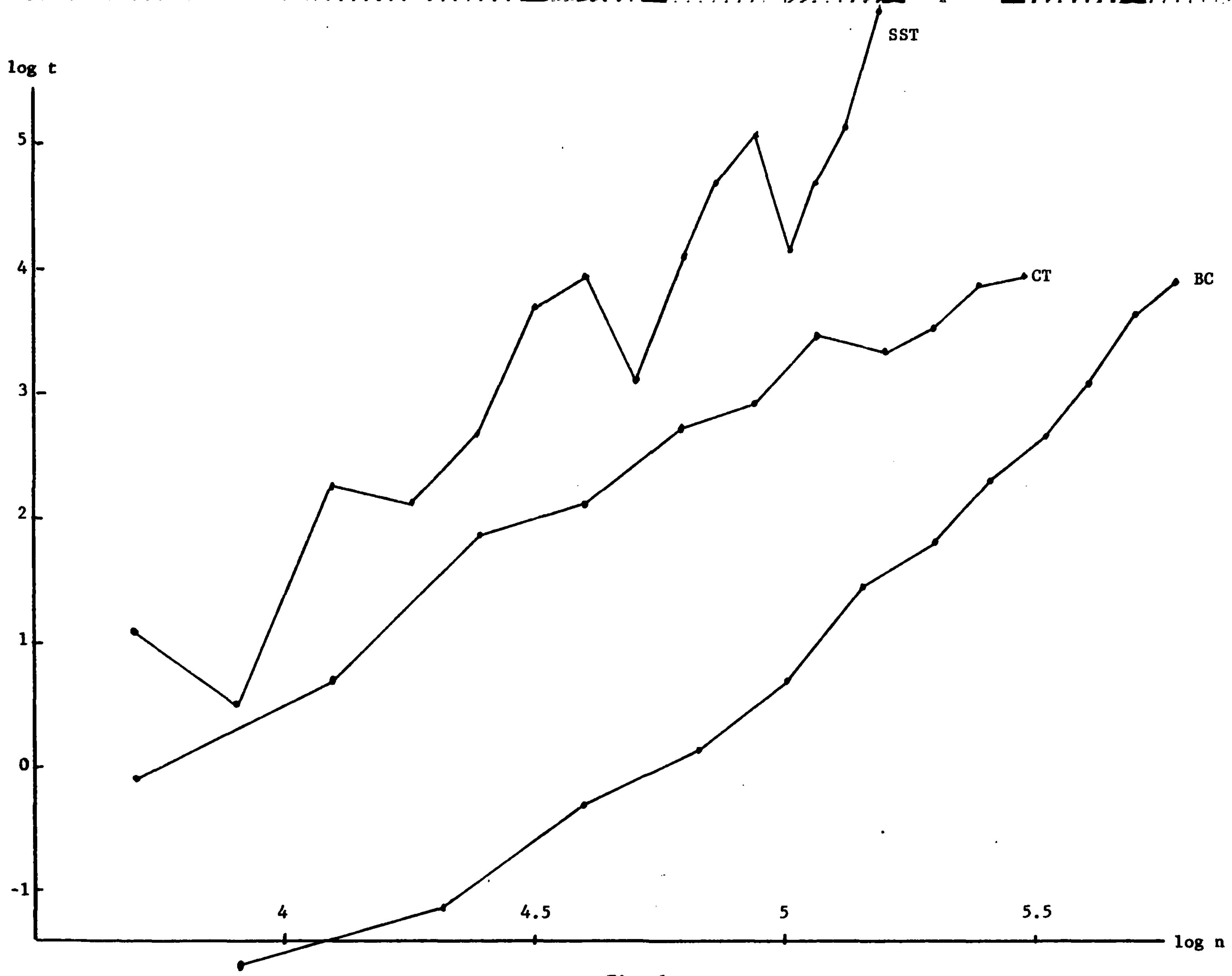


Fig. 1

<u>Table 3: Summary statistics for the twenty trials for each of eleven problem sizes for the CT algorithm</u>

Problem size (n_i)	Arithmetic mean ($\bar{t}_i$)	Geometric mean ($\bar{t}_i^G$)	Median ($t_i^{.5}$)	75th percentile ($t_i^{.75}$)	Standard deviation of t_{ij} (s_i)	Standard deviation of $\log(t_{ij})$ ($s_i^{\log}$)
40	0.92	0.79	0.82	1.20	0.48	0.578
60	2.22	1.76	2.07	3.04	1.59	.707
80	6.59	5.37	5.34	8.61	4.57	.656
100	10.41	8.28	7.38	15.22	7.04	.714
120	16.21	10.42	7.57	20.98	19.3	.893
140	18.70	14.20	11.07	27.33	15.6	.729
160	32.82	25.19	28.58	51.17	23.4	.776
180	28.76	19.44	25.85	39.85	25.5	.942
200	35.69	28.30	29.09	57.90	22.9	.739
220	46.71	30.31	23.33	59.10	50.9	.892
240	53.44	32.96	26.04	66.66	63.2	.946

Table 4: Simple correlations between pairs of summary statistics for solution times for the CT algorithm

	Arithmetic mean ($\bar{t}_i$)	Log of arithmetic mean ($\log[\bar{t}_i]$)	Geometric mean ($\bar{t}_i^G$)	Log of geometric mean ($\log[\bar{t}_i^G]$)	Median ($t_i^{.5}$)	75th percentile ($t_i^{.75}$)
($\log[\bar{t}_i]$)	0.886					
$\bar{t}_i^G$	.986	0.905				
$\log[\bar{t}_i^G]$	.875	.998	0.904			
$t_i^{.5}$	.896	.863	.941	0.872		
$t_i^{.75}$	.984	.901	.999	.899	0.948	
$s_i^{\log}$	.751	.775	.685	.734	.634	0.689

deviations of the natural logarithms of solution times, $s_i^{\log}$, increase with problem size, with a correlation of 0.78.

4. The Models

To represent computing time as a function of problem size, we investigated models of the form

$$(4.1) \qquad t_{ij} = f(n_i;\theta)\cdot e^{\epsilon_{ij}},$$

where n_i is problem size, t_{ij} is the solution time for the j-th trial for problem size n_i, θ is a vector of unknown parameters, and ϵ_{ij} is an error factor with mean $E(\epsilon_{ij})$ equal to zero and variance $E(\epsilon_{ij}^2)$ equal to σ_i^2, where E is the expectation operator. Taking natural logarithms of both sides of (4.1) yields

$$(4.2) \qquad \log t_{ij} = \log f(n_i;\theta)+\epsilon_{ij}\ ;$$

and denoting by N and M the number of problem sizes and trials, respectively, the problem of minimizing the sum of squared errors (known as the least squares problem) is

$$(4.3) \qquad \min_{\theta} \sum_{i=1}^{N} \sum_{j=1}^{M} [\log t_{ij} - \log f(n_i;\theta)]^2 .$$

At least for functions $f(n_i;\theta)$ which are linear in θ, least squares estimators are efficient (minimum variance) if the errors are homoscedastic (equal variance) and uncorrelated [$E(\epsilon_{ij}\epsilon_{hk}) = 0$ unless $i = h$ and $j = k$]. The multiplicative model (4.1) or (4.2) eliminates the worst of the correlation between the error variance and n_i, although even in this model some correlation remains for the Carpaneto-Toth algorithm (see Table 3).

We shall focus on three approximating functions: polynomial, super-polynomial (or log-exponential), and exponential, defined by

(4.4) $f(n_i) = \alpha' n_i^{\beta}$ or $\log f(n_i) = \alpha + \beta \log n_i$,

(4.5) $f(n_i) = \alpha' n_i^{\beta \log n_i}$ or $\log f(n_i) = \alpha + \beta \log^2 n_i$,

and

(4.6) $f(n_i) = \alpha' e^{\beta n_i}$ or $\log f(n_i) = \alpha + \beta n_i$

respectively, where $\alpha = \log \alpha'$ and, as before, log denotes the natural logarithm.

Since we have only arithmetic average solution times for each problem size for the SST and BC algorithms, we cannot solve the least squares problem (4.3). Averaging (4.2) over j produces

(4.7) $\log \bar{t}_i^G = \log f(n_i;\theta) + \bar{\epsilon}_i$,

where $\bar{t}_i^G$ is the geometric mean of the t_{ij}'s (or, equivalently, $\log \bar{t}_i^G$ is the arithmetic mean of the $\log t_{ij}$'s), and $\bar{\epsilon}_i$ is the arithmetic mean of the ϵ_{ij}'s, over all j. But since we cannot compute $\bar{t}_i^G$ for the SST and BC algorithms, we replace $\bar{t}_i^G$ with the arithmetic mean $\bar{t}_i$ for all three algorithms, and actually solve the least squares problem

(4.8) $$\min_{\theta} \sum_{i=1}^{N} [\log \bar{t}_i - \log f(n_i;\theta)]^2.$$

Since the correlation between $\bar{t}_i$ and $\bar{t}_i^G$ is 0.986 and the correlation between $\log(\bar{t}_i)$ and $\log(\bar{t}_i^G)$ is 0.998 for the CT algorithm, this approximation should not affect our results importantly.

Discriminating among models is difficult over the range of problem sizes spanned by our data, especially with the small number of observations available. Table 5 displays the simple correlations between the predicted solution times for the sample problem sizes for each pair of models (polynomial, superpolynomial, exponential) for each of the three algorithms (SST, CT, BC). The smallest correlation is 0.968 between the polynomial and exponential models for the BC algorithm. These high correlations suggest why it is difficult to estimate which function best describes the relationship between problem size and solution time for each algorithm.

5. The Results

We fitted the solution-time data for each of the three algorithms to each of the three models. The results are as shown in Table 6.

Each of the models is linear in the parameters α and β (see (4.4)-(4.6)), and is of the general form $\log \bar{t}_i = \alpha + \beta x_i + \epsilon_i$, where $x_i = n_i$ for the exponential model, $x_i = \log n_i$ for the polynomial model, and $x_i = \log^2 n_i$ for the superpolynomial model, and where n_i denotes the number of cities as a measure of problem size, while log denotes the natural logarithm. The symbols $\hat{\alpha}$ and $\hat{\beta}$ stand for the least squares estimates of α and β, respectively, while $s_{\hat{\alpha}}$ and $s_{\hat{\beta}}$ denote the standard errors (or square roots of the estimates of the variances) of $\hat{\alpha}$ and $\hat{\beta}$, respectively. The smaller the standard error of an estimated parameter, the more precise is our estimate of that parameter.

$\bar{S}$ is the standard error of the regression, defined as

$$(5.1) \qquad \bar{S} = \left(\frac{1}{N-p} \sum_{i=1}^{N} \hat{\epsilon}_i^2 \right)^{1/2} ,$$

Table 5: Simple correlations between least squares estimated solution times evaluated at sample problem sizes for indicated pairs of models for the SST, CT and BC algorithms

	Polynomial, superpolynomial	Polynomial, exponential	Superpolynomial, exponential
SST	0.999	0.980	0.988
CT	0.998	0.970	0.982
BC	0.998	0.968	0.980

Table 6: Least squares estimates of the model parameters, and associated statistics

	$\hat{\alpha}$ ($s_{\hat{\alpha}}$)	$\hat{\beta}$ ($s_{\hat{\beta}}$)	$\bar{S}$	$\bar{R}^2$	DW (Signif)
SST					
Polynomial	-11.39 (1.37)	3.243 (0.296)	0.5132	0.895	2.29
Superpolynomial	-4.187 (0.716)	0.361 (0.0328)	0.5108	0.896	2.26
Exponential	-0.0823 (0.406)	0.0331 (0.00343)	0.5745	0.868	1.74
CT					
Polynomial	-8.232 (0.531)	2.256 (0.110)	0.1980	0.977	1.34
Superpolynomial	-3.307 (0.373)	0.241 (0.0155)	0.2598	0.960	0.93 (0.05)
Exponential	-0.789 (0.369)	0.0140 (0.00215)	0.3462	0.929	1.52
BC					
Polynomial	-14.51 (0.953)	3.114 (0.186)	0.3607	0.962	0.83 (0.05)
Superpolynomial	-7.046 (0.367)	0.320 (0.0136)	0.2600	0.980	1.05
Exponential	-2.470 (0.137)	0.0205 (0.00066)	0.1985	0.989	0.96 (0.05)

Notes:

$\hat{\alpha},\hat{\beta}$	= least squares estimates of α,β
$s_{\hat{\alpha}},s_{\hat{\beta}}$	= standard error of $\hat{\alpha},\hat{\beta}$
$\bar{S}$	= standard error of regression
$\bar{R}^2$	= squared (adjusted) coefficient of multiple correlation
DW	= Durbin-Watson statistic
Signif	= level of significance

where p is the dimensionality of θ (here p = 2, since $\theta = (\alpha,\beta)$, and $\hat{\epsilon}_i = \log \bar{t}_i - f(n_i;\hat{\theta})$. $\bar{S}$ is a measure of the average distance of the data points from the regression line.

$\bar{R}^2$ is the (adjusted) squared multiple correlation coefficient (also called coefficient of (multiple) determination), defined as

$$(5.2) \qquad \bar{R}^2 = 1-(\bar{S}^2/\bar{S}_y^2),$$

where

$$\bar{S}_y^2 = \frac{1}{N-1}\sum_{i=1}^{N}(y_i-\bar{y})^2 ,$$

and where y_i denotes $\log \bar{t}_i$, while $\bar{y}$ is the arithmetic mean of the y_i's. $\bar{R}^2$ is the fraction of the variance of the $\log \bar{t}_i$'s explained by the regression. The closer to 1 is the coefficient of determination, and the closer to 0 is the standard error of the regression for a certain model, the better that model fits the data.

DW stands for the Durbin-Watson statistic, defined as

$$(5.3) \qquad DW = \frac{\sum_{i=2}^{N}(\hat{\epsilon}_i-\hat{\epsilon}_{i-1})^2}{\sum_{i=1}^{N}\hat{\epsilon}_i^2} .$$

It is a measure of the association of "adjacent" residuals. If the residuals are uncorrelated, DW = 2; if they are perfectly positively correlated, DW = 0; and if they are perfectly negatively correlated, DW = 4. In our context, a Durbin-Watson statistic significantly less than 2 for a certain model indicates that the model systematically overestimates and underestimates ranges of the curve described by the data, whereas a DW statistic significantly greater than 2 indicates frequent alternations of over- and under-estimates.

For further details on the statistical concepts used here and in the rest of the paper, the reader is referred to [8] and [11].

Based on the standard error of the regression and the coefficient of determination, none of the three models fits the data of the SST algorithm very well, although the polynomial and superpolynomial models fit them slightly better than the exponential one. The performance of the CT algorithm is best described by the polynomial function, with the superpolynomial a close second, and the exponential third. The exponential function fits best the BC data, with the superpolynomial a close second and the polynomial third.

Although the exponential function fits the BC data better than the other two functions, its DW statistic is nevertheless significant at the 5% level, i.e., even this "best" function systematically fails to capture the curvature of the "true" function behind the data (the 5% significance level means that the chances of getting a value as different from 2 as 0.96 if in fact the errors are uncorrelated, are less than one in 20). The DW statistics for the other "best" models do not significantly differ from 2.

More significant than the differences in model rankings for the three algorithms is the relative closeness of the fit for all three models, for each of the algorithms. Although the CT algorithm is best described by the polynomial model and the BC algorithm by the exponential model, note that both the exponential model for the CT algorithm and the polynomial model for the BC algorithm fit the data considerably better than any of the models for the SST algorithm.

Since we are particularly interested in the asymptotic behavior of these algorithms, it is important that there not be any systematic discrepancies between the actual data points and their estimated values for the larger problem sizes. For example, the model $y_i = \alpha + \beta x_i + \epsilon_i$ will fit data generated by $y_i = x_i^2$ very well over broad ranges of x_i; but the model will tend to underestimate y_i for small and large values of x_i and overestimate y_i for intermediate values of x_i. One clue in this case would be a DW statistic near zero. However, one cannot count on the DW statistic in a more complicated situation.

There is a simple procedure for testing whether the data for small and large problem sizes can be described adequately by the same model. Partition the data into two subsets: $I_1 = \{1,2,\ldots,[\frac{N}{2}]\}$, $I_2 = \{N + 1 - [\frac{N}{2}], N + 2 - [\frac{N}{2}], \ldots, N\}$, where $[\frac{N}{2}]$ is the greatest integer in $\frac{N}{2}$. Estimate the model for each subset of the data by least squares and let S_k^2 be the sum of squared residuals (denoted $\hat{\varepsilon}_i^k$) for the model associated with I_k:

$$(5.4) \qquad S_k^2 = \sum_{i \varepsilon I_k} (\hat{\varepsilon}_i^k)^2.$$

Let $I = I_1 \cup I_2$ and define $S^2 = \Sigma_{i\varepsilon I}\hat{\varepsilon}_i^2$. (Note that for N odd the set I excludes the middle observation.) Then the statistic

$$(5.5) \qquad u = \frac{S^2 - (S_1^2 + S_2^2)}{S_1^2 + S_2^2} \cdot \frac{2[\frac{N}{2}] - 4}{2}$$

has the F (Fisher's) distribution with 2 degrees of freedom in the numerator and $2[\frac{N}{2}] - 4$ degrees of freedom in the denominator. The larger this statistic is, the less likely it is that the data for I_1 and I_2 can be described by the same model.

The legitimacy of this test depends, inter alia, on the homoscedasticity of the errors ε_i. If we hypothesize that $\mathrm{var}(\varepsilon_i) = \sigma_k^2$, $i \in I_k$, then the statistic

$$(5.6) \qquad \omega = S_1^2/S_2^2$$

can be used to test the null hypothesis H_0: $\sigma_1^2 = \sigma_2^2$ against the alternative hypothesis H_a: $\sigma_1^2 \neq \sigma_2^2$. The statistic ω has the F distribution with $[\frac{N}{2}] - 2$ degrees of freedom in the numerator and denominator. The null hypothesis is rejected if ω is sufficiently different from 1 (the symmetry occurs because if H_0 is false, either $\sigma_1^2 > \sigma_2^2$ or $\sigma_2^2 > \sigma_1^2$). These tests were performed with the results shown in Table 7.

Table 7. Tests for homoscedasticity (ω) and model stability (u)*

Algorithm / Model	SST		CT		BC	
	ω	u	ω	u	ω	u
Polynomial	1.036	0.145	0.910	6.570 (0.05)	8.277 (0.1)	14.436 (0.005)
Superpolynomial	0.959	0.530	1.067	12.647 (0.01)	6.499 (0.1)	7.674 (0.025)
Exponential	0.815	3.816 (0.1)	4.139	25.611 (0.005)	1.926	13.914 (0.005)

* The numbers in parenthesis represent significance levels.

Only the polynomial and superpolynomial models for BC fail the test for homoscedasticity, which they fail only at the 10 percent significance level. To perform the stability tests for these two models we multiplied the first six observations by S_2/S_1, thereby insuring the equality of the residual variances for the two regressions based on I_1 and I_2.

Only the polynomial and superpolynomial models for SST, which are the better-fitting models for that algorithm, pass the model stability test. Stability of all the models for BC and CT is rejected. The problem is either that the asymptotic behavior of the BC and CT algorithms is not well described by any of the models, or that low-order effects confound our results because the samples do not include enough sufficiently large problem sizes.

Because the models generally fail the stability tests, the results for larger problem sizes may be better predictors of asymptotic behavior than are the results for all observations. Tables 8, 9, and 10 present the estimates for SST, CT, and BC, respectively, for the three models based on data for the smaller half and larger half of the problem sizes. These Tables dramatize why the models for BC and CT do not survive the stability tests. While the exponential model remains best for each half of the BC data, the exponent declines about thirty percent, from 0.0245 to 0.0170 (and the constant term increases from -2.913 to -1.568) from the first to the second half. These changes are far greater than could be expected from sampling error (as predicted by the standard errors of $\hat{\alpha}$ and $\hat{\beta}$), which is why the model stability test fails. The two functions predict equal solution times for problems roughly of size 179, which lies between the subsets of smaller and larger problem sizes on which the estimates are based. Overall, there is little difference in the within-sample predictive ability of the three models (as judged by $\bar{S}$).

The polynomial model remains best for smaller problem sizes for the Carpaneto-Toth algorithm, with the superpolynomial model a close contender. For the larger problem sizes, the exponential model is best, although there is little to choose among the three models. For all three models, the coefficient of the problem size explanatory variable decreases from the first (smaller problems) to the second (larger problems) regression. That is, the logarithms of solution times for smaller problems are more sensitive to problem size than are the logs of solution times for larger problems.

For the SST algorithm, the exponential model, which fits the entire data set worst, fits the two subsets best, although the differences among models are not great, especially for the larger problem sizes. Again, the sensitivity of log solution times to problem size declines for all three models, dramatically for the exponential.

Table 8. Least squares estimates for the SST algorithm.

	$\hat{\alpha}$ $(s_{\hat{\alpha}})$	$\hat{\beta}$ $(s_{\hat{\beta}})$	$\bar{S}$	$\bar{R}^2$	DW (Signif)
$40 \leq n \leq 100$					
Polynomial	-12.71 (2.77)	3.578 (0.657)	0.5276	0.827	2.72
Superpolynomial	-5.377 (1.358)	0.434 (0.0757)	0.5047	0.841	2.86
Exponential	-1.553 (0.626)	0.0554 (0.00860)	0.4550	0.871	3.22
$120 \leq n \leq 180$					
Polynomial	-11.36 (7.26)	3.243 (1.451)	0.5183	0.400	1.63
Superpolynomial	-3.312 (3.625)	0.327 (0.145)	0.5155	0.406	1.64
Exponential	1.493 (1.442)	0.0225 (0.00953)	0.5041	0.432	1.68

Notes: see Table 6.

Table 9. Least squares estimates for the CT algorithm.

	$\hat{\alpha}$ $(s_{\hat{\alpha}})$	$\hat{\beta}$ $(s_{\hat{\beta}})$	$\bar{S}$	$\bar{R}^2$	DW (Signif)
$40 \leq n \leq 120$					
Polynomial	-10.07 (0.661)	2.693 (0.153)	0.1325	0.987	3.10
Superpolynomial	-4.408 (0.365)	0.318 (0.0192)	0.1409	0.986	2.59
Exponential	-1.370 (0.345)	0.0364 (0.00407)	0.2574	0.952	1.55
$160 \leq n \leq 240$					
Polynomial	-3.791 (2.293)	1.407 (0.434)	0.1389	0.704	1.93
Superpolynomial	-0.0992 (1.131)	0.134 (0.0404)	0.1365	0.715	1.95
Exponential	2.189 (0.404)	0.00730 (0.00200)	0.1265	0.755	2.03

Notes: see Table 6.

Table 10. Least squares estimates for the BC algorithm.

	$\hat{\alpha}$ $(s_{\hat{\alpha}})$	$\hat{\beta}$ $(s_{\hat{\beta}})$	$\bar{S}$	$\bar{R}^2$	DW (Signif)
$50 \leq n \leq 175$					
Polynomial	-11.33 (1.29)	2.407 (0.277)	0.2879	0.937	1.72
Superpolynomial	-5.954 (0.563)	0.267 (0.0255)	0.2415	0.956	1.89
Exponential	-2.913 (0.138)	0.0245 (0.00115)	0.1203	0.989	3.15
$200 \leq n \leq 325$					
Polynomial	-21.43 (1.37)	4.379 (0.247)	0.1001	0.984	2.10
Superpolynomial	-9.319 (0.652)	0.395 (0.0211)	0.0947	0.986	2.30
Exponential	-1.568 (0.221)	0.0170 (0.00083)	0.0867	0.988	2.95

Notes: see Table 6.

In summary, these statistics do not offer much basis for deciding which of the three models describes best the performance of the three TSP algorithms. They nevertheless convey the important information that the best fitting polynomial functions have exponent around 3 (for the polynomial model, $2.2 \leq \beta \leq 3.2$), while the best fitting exponential functions have base around 1.02 (for the exponential model, the base b satisfies $e^{0.014} \leq b \leq e^{0.033}$). All three approximating functions for all three algorithms are shown in Table 11.

Table 11. Best Approximating Functions

Algorithm / Function type	SST $40 \leq n \leq 180$	CT $40 \leq n \leq 240$	BC $50 \leq n \leq 325$
Polynomial	$1.13 \times 10^{-5} \times n^{3.243}$	$0.27 \times 10^{-3} \times n^{2.256}$	$0.5 \times 10^{-6} \times n^{3.114}$
Super-polynomial	$0.15 \times 10^{-1} \times n^{0.361 \log n}$	$0.37 \times 10^{-1} \times n^{0.241 \log n}$	$0.87 \times 10^{-3} \times n^{0.320 \log n}$
Exponential	$0.92 \times e^{0.0331 n}$	$0.45 \times e^{0.014 n}$	$0.85 \times 10^{-1} \times e^{0.0205 n}$

References

[1] Balas, E. and Christofides, N., "A Restricted Langrangian Approach to the Traveling Salesman Problem," Mathematical Programming, 21, 1981, 19-46.

[2] Balas, E. and Toth, P., "Branch and Bound Methods for the Traveling Salesman Problems," MSRR No. 488, Carnegie-Mellon University, March, 1983.

[3] Bellmore, M. and Malone, J. C., "Pathology of Traveling Salesman Subtour Elimination Algorithms," Operations Research, 19, 1971, 278-307.

[4] Carpaneto, G. and Toth, P., "Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem," Management Science, 26, 1980, 736-743.

[5] Dantzig, G. B., Fulkerson, D. R. and Johnson, S. M., "Solution of a Large Traveling Salesman Problem," Operations Research, 2, 1954, 393-410.

[6] Hall, M., Combinatorial Theory, Blaisdell, 1967.

[7] Lenstra, J. K. and Rinnooy Kan, A. H. G., "On the Expected Performance of Branch and Bound Algorithms," Operations Research, 26, 1978, 347-350.

[8] Mood, A. M. and Graybill, F. M., Introduction to the Theory of Statistics, Second Edition, McGraw-Hill, 1963.

[9] Panayiotopoulos, J.-C., "Probabilistic Analysis of Solving the Assignment Problem for the Travelling Salesman Problem," European Journal of Operational Research, 9, 1982, 77-82.

[10] Smith, T. H. C., Srinivasan, V. and Thompson, G. L., "Computational Performance of Three Subtour Elimination Algorithms for Solving Asymmetric Traveling Salesman Problems," Annals of Discrete Mathematics, 1, 1977, 495-506.

[11] Theil, H., Principles of Econometrics, Wiley, 1971.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MSRR 501	2. GOVT ACCESSION NO. AD-A145786	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) STATISTICAL ANALYSIS OF SOME TRAVELING SALESMAN ALGORITHMS		5. TYPE OF REPORT & PERIOD COVERED Technical Report June 1984
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Egon Balas, Timothy W. McGuire, Paolo Toth		8. CONTRACT OR GRANT NUMBER(s) N00014-82-K-0329
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie Mellon University, Graduate School of Industrial Administration, Pittsburgh, Pennsylvania 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 047-607
11. CONTROLLING OFFICE NAME AND ADDRESS Personnel and Training Research Program, Office of Naval Research (Code 434), Arlington, Virginia 22217		12. REPORT DATE June 1984
		13. NUMBER OF PAGES 30
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

traveling salesman problem, branch and bound, algorithm performance, statistical analysis

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper reports the results of a statistical analysis of the performance of three branch and bound algorithms for the general (asymmetric) traveling salesman problem on randomly generated test problems with up to 325 cities. Three types of functions, polynomial, superpolynomial (log-exponential) and exponential, were fitted to the performance data of each of the algorithms by least squares techniques. The three functions describe almost equally well the behavior of the algorithms in the range of problem sizes examined.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)